Verification, Validation, and Failures of Uber ATG Self-Driving Car

By Ian Wilhite

ITDE 401-900

November, 25h, 2025

Table of Contents

1.0 Introduction and Need Statement

Autonomous vehicles promise to reduce human error by embedding situational awareness, decision-making, and control into a coordinated machine-intelligence framework to minimize the human error causing 94% of roadway crashes (*National Highway Traffic Safety Administration, 2017*). The overall need for such systems is clear: provide safe, reliable, and scalable autonomous transport that can coexist with human drivers and pedestrians. This study examines how structured systems engineering methods could have prevented such failures.

In March 2018, Uber's Advanced Technologies Group (ATG) self-driving SUV struck and killed pedestrian Elaine Herzberg in Tempe, Arizona *(NTSB, 2019)*. Within the perception-to-braking control chain, a flaw in classification and decision logic went unidentified during design verification, and was deployed during a validation deployment. This analysis is needed to evaluate how lapses in verification and validation can cause system-level failure, highlighting the importance of structured systems-engineering design processes. According to the National Highway Traffic Safety Administration (NHTSA), AV developers aim to reduce this figure, but the 2018 Tempe collision highlighted that algorithmic errors can be just as fatal when system-level oversight fails.

This paper applies systems-engineering concepts to examine how the product's need, conceptualization, and functional architecture evolved, where the design process broke down, and at what stage the fatal flaw should have been detected through systematic verification and peer review. To understand how these failures could have been prevented, this report will examine the systems engineering processes that should have guided the development of the autonomous system.

2.0 Design Process

The conceptualization phase defines how a system's purpose is translated into actionable requirements for verification. Systems engineering begins with a clear statement of need, mission, and stakeholder expectation. From this need, the engineering requirements are derived to govern the structure of the system.

This solution-biased framing led to requirements written around performance milestones rather than traceable performance metrics. A rigorous conceptualization process would have employed a structured V-model or model-based systems-engineering (MBSE) approach, emphasizing traceability between stakeholder needs, derived requirements, and verification evidence.

2.1 Deriving Stakeholder Needs

At the conceptual stage, Uber ATG defined the mission: develop a self-driving rideshare platform capable of fully autonomous operation in mixed-traffic urban environments. Stakeholders included passengers (comfort, safety), regulators (compliance), corporate sponsors (schedule and cost), and the general public (trust in emerging technology).

At the conceptual stage, Uber's Advanced Technologies Group (ATG) defined its mission objective. Translating this vision into actionable design requirements required identifying and reconciling the diverse needs of multiple stakeholders with distinct priorities, performance metrics, and risk tolerances.

Passengers represented the system's end users. Their needs centered on personal safety, ride comfort. From an engineering standpoint, this translated into quantitative requirements such as maintaining lateral acceleration below 2 m/s^2 for comfort, achieving redundant perception for collision avoidance, and ensuring high system uptime.

Federal regulators within the National Highway Traffic Safety Administration (NHTSA) and local Departments of Transportation sought compliance with safety frameworks for automated driving systems. Their needs involved verifiable adherence to standards, incident traceability, and data retention. Regulators also required that Uber ATG demonstrate structured risk assessment and validation plans before public-road testing, establishing the guidelines for public safety.

For corporate sponsors, success was measured in development speed, cost efficiency, and competitive advantage in the rideshare market. Their needs emphasized rapid deployment, reduced testing cycle times, and scalable software integration across the fleet. However, these priorities often create tension with engineering and regulatory needs; aggressive timelines risk compressing verification and validation (V&V) activities, as demonstrated by the 2018 incident.

Recognizing and quantifying these stakeholder needs formed the foundation for Uber ATG's system-level requirements and should have guided all subsequent design, verification, and validation decisions.

2.2 Problem Statement

Rather than beginning with an engineering problem, Uber's early framing was solution-biased toward "achieving autonomous ride operation." A solution-neutral problem statement should have been:

> "Develop a vehicle control system that can perceive, predict, and respond to dynamic roadway hazards with reliability equal to or greater than that of a typical attentive human driver."

This statement would have anchored subsequent requirements around measurable performance rather than feature milestones. The mission as identified in the problem statement is the first step into any requirement flowdown, and a poor or vague mission could lead to unreasonable or incorrect research and development costs unrelated to consumer or stakeholder needs.

2.2 Design-Process Model

The ideal development process for safety-critical systems follows the V-model of systems engineering, which establishes a clear relationship between specification and verification at every level of design:

V-Model Phase	Intended Purpose	Application & Deviation
1. Concept of Operations (ConOps)	Define mission objectives, operating environment, and system boundaries. Establish stakeholder needs and safety goals before design begins.	The mission was defined in aspirational terms without strictly bounding the operational design domain.
2. System Requirements	Translate mission needs into measurable, testable performance metrics.	Requirements emphasized performance rather than quantitative safety targets.
3. High-Level Design	Derive subsystems to define data interfaces and dependencies.	Uber's architecture included these subsystems but lacked formally controlled interfaces.
4. Detailed Design & Implementation	Develop algorithms, hardware, and integration logic that meet subsystem specifications.	Neural-network classifiers and motion-prediction modules were implemented using limited datasets.
5. Verification & Validation (V&V)	Verify each component against its requirements then validate that the system meets needs.	Testing ended at module-level verification; end-to-end perception-to-braking validation was never performed.
6. Operations & Feedback	Monitor field performance, collect operational data, and integrate feedback into design updates.	Field testing did not prioritize safety analysis.

Table 2.2a: Design Process Model

By not bounding the operational design domain (ODD) in the Concept of Operations phase, the team permitted test vehicles to operate in unstructured, mixed-traffic environments that exceeded the maturity of their perception and decision-making algorithms. This omission prevented engineers from defining realistic environmental assumptions that would later compromise classification accuracy in night-time testing.

At the high-level and detailed design stages, the lack of controlled subsystem interfaces and data traceability led to inconsistent assumptions between the perception and motion-planning teams. For example, the perception module could classify objects using limited datasets while the motion-planning logic relied on confidence values never formally verified against ground truth. This architectural disconnect meant that false classifications fed into control decisions, and could execute incorrect behavior.

By compressing the verification and validation (V&V) phases into agile field testing, Uber ATG blurred the line between experimentation and qualification. Unverified software iterations were deployed on public roads before completing end-to-end Software-in-the-loop (SITL) simulation and validation.

Finally, the operations and feedback phase failed to close the loop that the V-model demands. Post-test data were collected, but safety insights were not systematically fed back into design revisions or hazard-tracking databases. Without that continuous improvement cycle, the product's safety performance plateaued instead of converging toward effective robustness.

3.0 System Architecture

The autonomy system deployed by Uber ATG was typical with common industry trends, consisting of layered software elements to transform sensor data into actuator commands. Each module executes a distinct function linked through time-synchronized data buses, such that they may be tested and developed independently. These include perception, state estimation, mapping, path planning, trajectory optimization, and control. Figure 3.1a illustrates the simplified architecture of this autonomy stack.

3.1 Functional Diagram

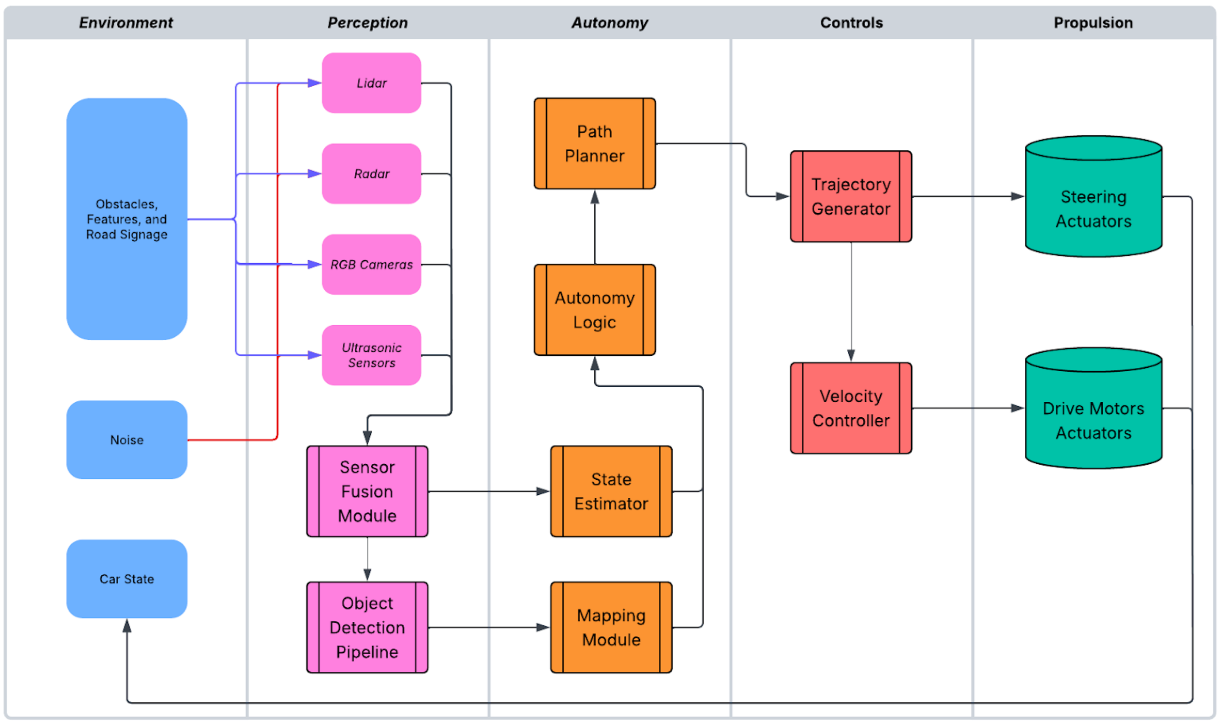

Figure 3.1a: Functional Architecture Diagram

Each colored column represents a functional subsystem, each bolded block represents a necessary function, each unbolded box represents a specific instrument or known feature. Information flows generally from right to left, as the environment is observed, the data is processed, features are identified, decisions are evaluated, controllers stabilize the system, and actuators affect the surrounding environment.

Uber's sensor suite combined lidar for geometry, radar for velocity, and camera for color data. However, the fusion module lacked redundancy. The lidar could have contradicted the “unknown object” label, yet the planner relied on a single unverified source, even when it received conflicting information.

Sensor data are fused in real time and passed through a neural-network classifier. The classifier outputs labeled objects with confidence scores and relative positions. These outputs feed a motion-prediction model that forecasts object trajectories. The local planner then computes an optimal vehicle path that minimizes risk and adheres to traffic laws, while the low-level controller executes commands at millisecond rates using proportional integral derivative (PID) or model-predictive control (MPC) control system algorithms. The heavy reliance on machine-learning based methods allow for the introduction and propagation of misclassifications.

In a fully validated design, a redundant safety monitor would continuously assess sensor health, possible reaction time, and feasible braking distance to trigger an emergency stop if confidence values fall below an established threshold. In Uber’s implementation at the time of the accident, this supervisory layer was partially implemented and, crucially, the Original Equipment Manufacturer (OEM) Automatic Emergency Breaking (AEB) system was disabled to avoid control conflicts, eliminating an independent safeguard.

3.2 Feature Translation

Function	Feature Implementation	Intended Performance
Perception	Lidar-based object detection & radar confirmation	Detect pedestrians ≥ 50 m ahead
Classification	Neural-network labeler	≥ 99 % correct pedestrian ID
Prediction	Motion model of detected object	Predict crossing within 2 s
Planning	Trajectory generator	Maintain < 300 ms delay
Control	Brake and steering commands	Achieve ≥ 0.6 g decel
Safety Layer	Volvo OEM AEB	Redundant independent stop function

Table 3.2a: Feature Translation Table

The system as delivered deviated from this functional intent when the OEM AEB was disabled and the perception classifier mis-labeled the pedestrian as an “unknown object,” preventing the planner from invoking braking.

4.0 Design Methodology in Practice

4.1 Requirements Development

Within the V-model, the requirements phase establishes the measurable criteria that drive design verification. Quantitative safety requirements (reaction time, detection distance) were not documented at the top level. Testing emphasized "autonomous-miles-driven" rather than verification coverage, leaving no measurable threshold for acceptable perception performance. The absence of quantitative limits on perception latency or braking distance prevented meaningful traceability between system requirements and industry standards like *ISO 26262: Functional Safety of Road Vehicles* (2018), making it impossible to prove hazard-mitigation compliance.

4.2 Concept Evaluation

During concept selection, engineers faced a trade-off between retaining the OEM AEB and avoiding command conflicts between the autonomy stack and Volvo's built-in controller. The decision to disable AEB lacked a documented trade study—an omission of the "concept evaluation" phase. This choice removed the only validated redundancy and converted a software bug into a single-point failure. This omission removed a validated safety layer and introduced a single-point failure to the perception module's decision logic directly commanded braking without independent arbitration or sanity-check logic.

4.3 Sources of Design Information

Training data originated mainly from daylight crosswalk scenarios. The lack of reference data of nighttime trials meant that the classification model was not trained to detect people at night, and therefore when presented with a new unknown, did not see the pedestrian. The source information (requirement 1f) therefore did not represent the full use case and operational design domain.

4.4 Verification and Validation Gap

The perception subsystem passed internal simulation tests but was never validated at system level against integrated braking response as described by *NASA-STD-7009: Model and Simulation Verification and Validation* (2016). Verification ended at module output rather than final behavior, meaning the disconnect between classification and control went unnoticed. Without scenario coverage analysis or integrated end-to-end testing, latent errors in object identification and tracking propagated into motion planning modules unchecked.

Industry practice for perception systems includes measuring precision and recall against a labeled dataset and performing scenario coverage analysis using Operational Design Domain (ODD) matrices. Uber’s internal test lacked this coverage, leading to untested edge cases such as low-contrast night pedestrians.

5.0 Project-Management Dimensions

5.1 Work Breakdown Structure

A proper Work Breakdown Structure (WBS) would define integration deliverables linking perception, planning, and controls. Uber's structure separated these teams with independent schedules and repositories. Integration relied on nightly software builds without cross-team verification, introducing uncontrolled configuration risk.

5.2 Risk Management

A disciplined systems-engineering process requires that technical and operational risks be continuously identified, ranked, and mitigated according to their potential impact on safety and mission success. In the Uber ATG program, this practice was largely informal, as hazards were logged in ad-hoc spreadsheets and anecdotal driver reports rather than in a controlled risk register or Failure Mode and Effects Analysis (FMEA). When test operators observed "phantom braking," management treated it as a nuisance to be suppressed rather than a signal of latent safety conflict between the autonomy stack and the Original Equipment Manufacturer (OEM) automatic emergency braking (AEB) system. The absence of a quantitative framework allowed subjective decisions to override objective hazard severity.

A proper FMEA would have revealed several single-point software and integration failures including pedestrian misclassification, AEB disablement, or asynchronous sensor timing. The table below summarizes high-risk items, with severity (S), occurrence (O), and detection (D) ranked on a 1–10 scale and combined into a Risk Priority Number (RPN = S × O × D). Items with an RPN of approximately 200 warrant corrective action before field deployment.

#	Failure Mode	Effect on System	Likely Cause(s)	Current Controls	S	O	D	RPN	Recommended Actions (Mitigation)	Owner
1	Incorrect perception	Collision risk	Low-light gap; missing night data	Module-level tests	10	7	6	**420**	Expand night datasets; require precision/recall ≥ target at night	Perception
2	AEB disabled	Loss of last-resort stop	Avoiding command conflicts	Safety driver	10	5	9	**450**	Re-enable OEM AEB; add arbiter with priority logic	Systems & Safety
3	Time sync drift across sensors	Incoherent tracks; late braking	Unsynchronized clocks; latency	NTP; nightly builds	8	6	7	**336**	PTP/1588 sync; time offset estimator in fusion; CI test that injects drift	Platform/ Fusion
4	Unverified sensors	Planner ignores contradictions	No cross-sensor checks	Basic thresholding	9	5	7	**315**	Cross-modality verification; bayesian estimation methods	Fusion Lead
5	Delayed braking	Delays on strange objects	Aggressive thresholds	Manual tuning	9	6	6	**324**	Conservative low-confidence policy	Planning Lead
6	Safety driver distracted	Delayed human intervention	HMI load; weak enforcement	Basic driver policies	9	6	5	**270**	Strict no-device policy; 2-person for test ops	Test Ops
7	ODD mismatch	Unhandled scenario	Coverage gap in test matrix	Generic road tests	7	6	6	**252**	ODD catalog + scenario coverage targets	V&V
8	Sensor occlusion	Degradation unnoticed	No health checks or fallback	Manual checks	8	5	6	**240**	On-line sensor self-check; degraded-mode behavior	Hardware/ Perception
9	Excess speed	Cannot stop within visible range	Max speed not tied to visibility	Speed limits	8	5	6	**240**	Dynamic speed policy; prove stopping-sight compliance	Control/ Planning

Table 5.2a: Failure Mode and Effects Analysis (FMEA) Table

5.3 Project Tracking and Peer Review

Project metrics tracked mileage and progress but not hazard-response success rate. Independent peer review as advised by NASA and NSPE was totally absent. A stage-gate peer review before enabling public-road testing would have forced verification evidence and risk closure. Many of the failures in verification could have been prevented through thorough design reviews and external evaluations before progressing into validation.

6.0 Ethics of Design

6.1 Societal Impacts

The immediate result was the death of a pedestrian; the secondary effect was a collapse in public trust in AV technology. Regulators tightened testing standards, delaying commercial rollout across the automotive industry. The design's failure thus affected not only Uber's program but the societal acceptance of autonomy. Self-driving programs at other companies including Tesla and Waymo have been met with stringent regulation and safety requirements, due in part to the Uber ATG incident.

6.2 Ethical Responsibility

Under the NSPE Code of Ethics, engineers must prioritize "holding paramount the safety, health, and welfare of the public". Testing unvalidated autonomy on open roads violated this principle. The decision to disable an independent safety feature for convenience or accelerated validation is emblematic of "normalization of deviance".

6.3 Cultural Context

U.S. regulatory culture favored innovation first, regulation later. In the EU, comparable trials would require a safety case demonstrating compliance with ISO 26262 and UNECE regulations. Cultural leniency effectively transferred risk from corporation to public, magnifying the ethical breach.

7.0 UBER ATG Case Study

By mapping the incident through the systems-engineering hierarchy:

Level	Expected Output	Actual Issue	Confirmation Phase
Concept of Operations	Define mission, environment, constraints	Nighttime, mid-block pedestrians not identified as use case	Concept Development
System Requirements	Quantitative detection/braking metrics	No defined performance thresholds	Requirements Definition
Functional Decomposition	Consistent interface timing	Asynchronous perception/planning rates	Preliminary Design
Subsystem Design	Reliable object classification	Mis-classification of pedestrian	Detailed Design
Integration & Test	End-to-end validation	AEB disabled; integration unverified	Verification & Validation

Table 7.0a: V-Model deviations table

The root cause lay within the perception subsystem, but the missed detection of that flaw occurred during requirements and integration verification, where a disciplined review should have flagged the mismatch between system need and subsystem capability before moving to validation.

8.0 Conclusion

The Uber ATG collision demonstrates how a single software mis-classification became a fatal event through systemic failures in design methodology and project management. Autonomous rideshare vehicles serve a legitimate societal need, yet an implementation contradicting the principles of systems engineering led to a severe accident.

- Systems should be verified before deployment to validation
- System architecture should be designed around need not tools
- System design processes bring stability and safety

The tragedy in Tempe is a lesson in engineering accountability. Systems engineering matters because it transforms innovation into a process surrounded by reliability. As autonomy advances, structured verification and validation, ethical responsibility, and transparent safety assurance must become the defining features of engineering practice.

References

1. **National Transportation Safety Board (NTSB).** *Collision Between Vehicle Controlled by Developmental Automated Driving System and Pedestrian, Tempe, Arizona, March 18, 2018.* Highway Accident Report NTSB/HAR-19/03, Washington, D.C., 2019.

2. **International Organization for Standardization (ISO).** *ISO 26262: Road Vehicles — Functional Safety.* Geneva, Switzerland: ISO, 2018.

3. **NASA. (2016)**. *NASA-STD-7009: Standard for Models and Simulations. National Aeronautics and Space Administration.*

4. **National Society of Professional Engineers (NSPE).** *NSPE Code of Ethics for Engineers.* Alexandria, VA, 2024 Edition.

5. **Society of Automotive Engineers (SAE).** *J3016: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles.* Warrendale, PA, 2021.